Amendments to the Specification

Please replace the paragraph that begins on Page 5, line 12 and carries over to Page 6, line 10 with the following marked-up replacement paragraph:

-- Most implementations of routing tables today use radix trees. Radix trees require a significant amount of programming logic, and expenditure of a significant amount of computing time in traversing the trees to find a particular route. Furthermore, radix trees cannot exploit a multi-processor (MP) approach wherein the computing task is shared among processors. Another existing technique is use of sorted linked lists. Linked lists have well-known performance problems, and are also not MP-exploitable. Some existing implementations use hash tables along with sorted linked lists. This approach provides performance which is significantly better than linked lists alone, but still does not provide an optimal (nor an MP-exploitable) solution. A technique designated "DIR-24-8-BASIC" was proposed by Pankaj Gupta et al. at Infocomm 1998, where two separate routing tables are used: one table for routes which are less than 25 bits long, and a different table for routes which are 25 bits or longer. This technique, however, assumes that most routes have prefixes of 24 bits or less and is therefore thought by the inventor of the present invention to have a rather narrow focus. (A copy of this conference paper ~~may be found~~ was published on the Internet at a web page of Stanford University.) ~~address http://www-cs-students.stanford.edu/~pankaj/paps/Infocom98.pdf.)~~ A technique known as "Multi-Protocol Label Switching" (MPLS) has also been proposed, where this technique would replace the longest-prefix match approach with a simple direct lookup. This approach, however, would require adoption of new protocol standards, and thus is not easily nor quickly adaptable into the established distributing computing infrastructure. --

Please replace the paragraph on Page 15, lines 3 - 9 with the following marked-up replacement paragraph:

-- Note that the preferred embodiment uses 8-byte entries, which are convenient when using CDS (Compare Double and Swap) operations in an MVS (Multiple Virtual Storage) system. Alternatively, array entries of other lengths can be used without deviating from the scope of the present invention, and those entries may be divided in ways other than using 2 equal-sized divisions. Furthermore, the array entries may contain information which is interpreted differently than that used to illustrate the operation of the preferred embodiment. These alternatives are within the scope of the present invention. --

Please replace the paragraph that begins on Page 16, line 14 and carries over to Page 17, line 17 with the following marked-up replacement paragraph:

-- Suppose a routing table entry is to be added for the route 9.67.96.0, using the network mask 255.255.255.0. This route is referred to as "R1" in Figs. 3 and 4, and its mask is referred to as "G1". The value of mask G1 indicates that the first 3 of 4 address components are significant when determining routing using route R1. Because there is more information than just the first address component (having the value 9) in this routing table entry, the more flag 302 associated with index entry 9 (shown as element 301) in the first array 300 is set on. Similarly, the more flag 322 associated with index entry 67 (element 321) in the second array 320 is set on. Because there is no further significant information, according to mask G1, after the third address component, the [[mask]] more flag for the value 96 (shown at 342 for index entry 341) is initially set to off. (Note that the value of more flag 342 is shown in Fig. 3 as being set on: this value is changed

from its initial setting when processing the next example described below.) The routing table information associated with this route is stored in a control block 344, which is pointed to by the pointer field 343. In the preferred embodiment, a "use count" is associated with each control block, indicating how many different routes use the information in the control block. Because the G1 mask is a specific value for all 24 bits which comprise the first 3 address components, only 1 route uses this R1 information and thus the use count 345 is set to 1. With this R1 entry in the routing table, a route lookup for IP address 9.67.96.x (where "x" represents an arbitrary value) using a mask length of 24 bits would then access element 301 in array 300; it would see from more flag 302 that there was more information to be learned by accessing element 321 in array 320; it would see from more flag 322 that array 340 should be checked; it would access array 340 at element 341; and it would then determine from more flag 342 (which would be set to off at this point) that there was no more information to be learned from accessing the last array 360. Thus, the routing information in control block 344 would then be retrieved by following pointer 343. --

Please replace the paragraph on Page 21, lines 4 - 13 with the following marked-up replacement paragraph:

-- As has been demonstrated, the present invention provides a novel technique for lookup [[up]] <u>of</u> IP addresses which is fast and efficient, and which requires a minimal amount of storage. Using this technique, lookup time is nearly constant, regardless of the value being looked up. This technique may be exploited using multiple processors to further increase lookup speed. The disclosed technique may be used equally well with class-specific IP version 4 32-bit addresses and with Classless Inter-Domain Routing (CIDR), and easily extends to use with IP version 6 128-bit

addresses. The technique capitalizes on the fact that the most specific searches are at an array associated with the final address component, and the least specific are at an array associated with the first address component. Furthermore, it exploits how IP addresses are subnetted, and how routes are added for various subnets/hosts for a particular organization. --